# DRAFT: Documentation of `ModelSolver` A `Python`-class for analyzing dynamic algebraic models

Magnus Kvåle Helliesen

March 20, 2023

**Abstract**

This article documents the `Python` class `ModelSolver`. The class lets the user define a model in terms of equations and endogenous variables. It contains methods to solve and analyze the model.

## 1 Background

### 1.1 The model

Suppose we have a model,

$$\begin{aligned} L_1(\mathbf{x}_t, \mathbf{z}_t) &= H_1(\mathbf{x}_t, \mathbf{z}_t), \\ L_2(\mathbf{x}_t, \mathbf{z}_t) &= H_2(\mathbf{x}_t, \mathbf{z}_t), \\ &\vdots \\ L_n(\mathbf{x}_t, \mathbf{z}_t) &= H_n(\mathbf{x}_t, \mathbf{z}_t), \end{aligned}$$

where $\mathbf{x}_t = (x_{1,t}, x_{2,t}, \ldots, x_{n,t})$ is a vector of endogenous variables, and $\mathbf{z}_t$ is a vector of exogenous variables and lags. The model can be re-written as

$$\mathbf{F}(\mathbf{x}_t, \mathbf{z}_t) = \begin{pmatrix} L_1(\mathbf{x}_t, \mathbf{z}_t) - H_1(\mathbf{x}_t, \mathbf{z}_t) \\ L_2(\mathbf{x}_t, \mathbf{z}_t) - H_2(\mathbf{x}_t, \mathbf{z}_t) \\ \vdots \\ L_n(\mathbf{x}_t, \mathbf{z}_t) - H_n(\mathbf{x}_t, \mathbf{z}_t) \end{pmatrix},$$

and the solution to the model is given by

$$\mathbf{F}(\mathbf{x}_t, \mathbf{z}_t) = \mathbf{0}.$$

On the face of it, this is a problem which the *Newton-Raphson*-algorithm handles well. The issue, however, is that $n$ might be quite large. In the Norwegian

Figure 1: Bipartite graph (BiGraph of model)

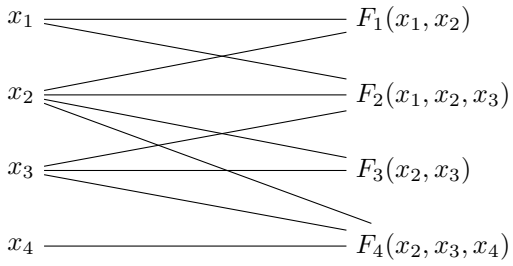
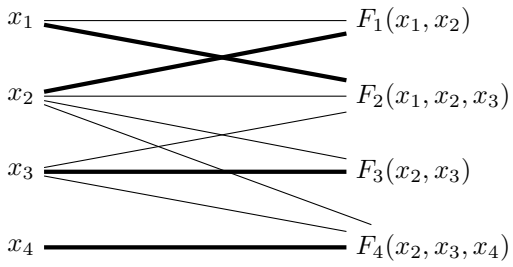



Figure 2: Maximum Bipartite Match (MBM) of BiGraph




national accounts, $n$ is somewhere between 15,000 and 16,000. Therefore, it is useful to analyze the system of equations before solving it in order to break it down into minimal simultaneous blocks that can be solved in a particular sequence.

## 1.2 Block analysis

In order to analyze and divide the model into blocks, we use results from *graph theory*.

First, we construct a *bipartite graph* (BiGraph) that connects endogenous variables with equations. This is illustrated in Figure 1 for an arbitrary model with 4 equations (we omit time subscripts for notational convenience).

Next, we apply *maximum bipartite matching* (MBM), which assigns each endogenous variable to *one and only one* equation. The MBM is arbitrary, and Figure 2 shows one possible solution.

We use Figure 2 to determine which endogenous variables impact which *other* endogenous variables. This is illustrated in Figure 3.

Next, we use Figure 3 to construct a *directed graph* (DiGraph) that shows how endogenous variables impact each other. The DiGraph is shown in Figure 4.

Finally, we find the *strong components* of the DiGraph, as shown in Figure

Figure 3: Graph of what endogenous variables impact what *other* endogenous variables

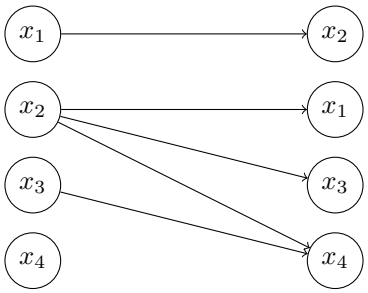



Figure 4: Directed graph (DiGraph) of what endogenous variables impact what *other* endogenous variables

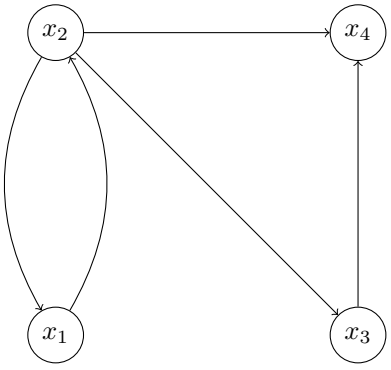


Figure 5: Condensation of DiGraph

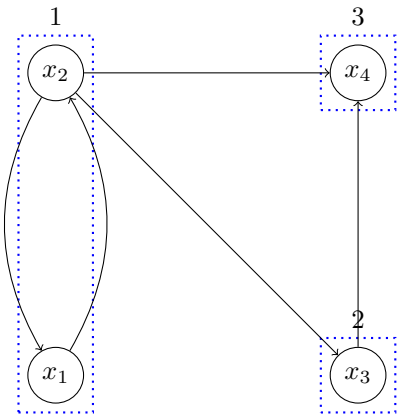
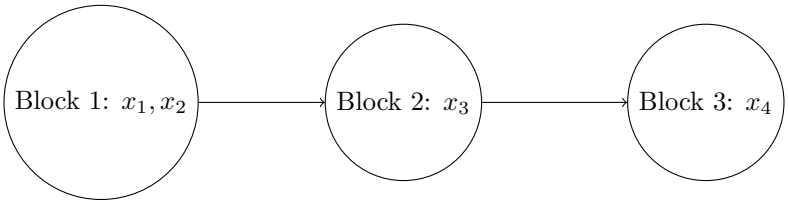



Figure 6: Condensed DiGraph




5. The strong components are nodes that are connected such that every node can be reached from every other (traversing the arrows). A *condensation* of the DiGraph is a (new) DiGraph with each node being the strong components (of the former). The condensation can be illustrated as in Figure 6. Each node of the condensation corresponds to a block of the model, and the arrows decide the sequence in which the blocks are to be solved.

In this example, $x_1$ and $x_2$ must be solved first in one simultaneous block. Next, $x_3$ is solved (taking $x_1$ and $x_2$ as given by the solution to block 1). Finally, $x_4$ is solved (taking $x_1$, $x_2$ and $x_3$ as given by the solutions to block 1 and 2).

## 1.3 Simulation code

With the blocks of the model given by the condensation of the DiGraph, we can generate *simulation code*. That is *either*

- a function that takes exogenous input and returns the endogenous value (if the block is a definition as described below), or
- a symbolic *objective function* and *Jacobian matrix* which are to be sent to a Newton-Raphson algorithm to be solved.

A block is a said to be a *definition* if and only if it 1) consists of one equation and 2) the endogenous variable is alone on the left hand side of the equation (and that endogenous variable does not show up on the right hand side of that equation).

### 1.4 Solution

If the block to be solved is not a definition, it is sent to a Newton-Raphson algorithm to be solved. The $k+1$st iteration of the Newton-Raphson algorithm is given by

$$\mathbf{x}_t^{(k+1)} = \mathbf{x}_t^{(k)} - \mathbf{J}_{\mathbf{F}}^{-1}(\mathbf{x}_t^{(k)}, \mathbf{z}_t)\mathbf{F}(\mathbf{x}_t^{(k)}, \mathbf{z}_t).$$

We stop if $\max\left(\left|\mathbf{x}_t^{(k+1)} - \mathbf{x}_t^{(k)}\right|\right) \leq \varepsilon$, where $\varepsilon$ is a tolerance level.

## 2 Examples of use

```
model = ModelSolver(equations, endogenous)
```

More to TBA.